Octopus: Anonymous and Secure DHT Lookup

Abstract

Distributed Hash Table (DHT) lookup is a core technique in structured peer-to-peer (P2P) networks. Its decentralized nature introduces security and privacy vulnerabilities for applications built on top of them; we thus set out to design a lookup mechanism achieving both security and anonymity, heretofore an open problem. We first explore the essential anonymity vulnerabilities in DHT lookups, and propose mechanisms to address them. We present Octopus, a novel anonymous and secure DHT lookup, which provides strong guarantees for both anonymity and security. We design a novel attacker identification mechanism to discover and remove misbehaving malicious nodes, thus severely limiting an adversary's ability to carry out active attacks. We analyze the security of Octopus by developing an event-based simulator to show that the attacker discovery mechanism can rapidly identify malicious nodes with low error rate. We calculate the anonymity of Octopus using probabilistic modeling and show that Octopus can achieve near-optimal anonymity guarantees: the anonymity provided by Octopus is 4-6 times better than that of any existing scheme (in terms of amount of leaked information). We evaluate Octopus's efficiency on Planetlab with 207 nodes and show that Octopus has reasonable lookup latency and manageable communication overhead.

1 Introduction

Structured peer-to-peer networks, such as Chord [34] or Kademlia [19], allow the creation of scalable distributed applications that can support millions of users. They have been used to build a number of successful applications, including P2P file sharing (Overnet, Kad, Vuze DHT[1]) and content distribution (CoralCDN [16]), and many others have been proposed, such as distributed file systems [30], anonymous communication systems [21, 27, 28, 20, 23], and online social networks [33, 8]. At the heart of these networks lies a distributed hash table (DHT) lookup mechanism that implements a decentralized key–value store. The DHT allows efficient storage and coordination among very large collections of nodes; however, its decentralized nature creates a number of security and privacy vulnerabilities. Because peers have to rely on other peers to determine the state of the network, malicious nodes could provide misinformation to disrupt and misdirect an honest user's lookups [36]. Likewise, nodes can profile the lookup activities of other nodes and learn what files or websites they are interested in or who their friends may be. Recent research shows that anonymous communication systems based on anonymity-deficient DHT lookups have severe vulnerabilities to information leak attacks [22, 38].

To address these issues, our goal is to design a lookup mechanism that is both secure and *anonymous*, i.e., it reveals no information about which nodes are looking up which values. To date, a lookup that satisfies both these properties has remained elusive. Several proposals for secure DHT lookups have been put forward [7, 27, 3, 17, 37, 26]; however, these schemes are not designed to preserve anonymity. For example, lookup keys are simply revealed during lookup queries, lookup initiators' identities are exposed due to contacting intermediate nodes directly, and using redundant lookups for security assurance accelerates information leaks [22]. NISAN [28] was a first attempt

[1] `http://www.vuze.com/`

at creating a secure and anonymous lookup, but it was shown to provide a very low amount of anonymity protection [38].

We therefore examine the fundamental challenges to providing both anonymity and security in DHT lookups and propose mechanisms to address them. We present Octopus, a novel DHT lookup that provides strong guarantees for both security and anonymity. Octopus uses three basic techniques to achieve its goals. First, it constructs an anonymous path, using Onion routing techniques [35], to relay lookup query messages while hiding their initiator. Second, it splits individual queries used in a lookup over multiple anonymous paths, and introduces dummy queries, to make it difficult for an adversary to learn the eventual target of a lookup. Third, it uses a novel attacker identification mechanism to discover and remove malicious nodes that launched active attacks; this mechanism provides a new dimension of defense to both active and passive attacks, since adversaries who use active attacks will compromise their ability to monitor traffic via passive attacks as their nodes will be ejected from the network.

We systematically evaluate the security and anonymity of Octopus by extensive analysis and simulations. We develop an event-based simulator to show that our attacker identification mechanism is capable of rapidly discovering malicious nodes with low error rate. For a network with 20% malicious nodes, Octopus can correctly identify all attacking nodes within 30 minutes. Unlike most previous works that only analytically evaluate the systems' anonymity, we use probabilistic modelling with the help of simulation to calculate the anonymity that is actually provided by the system, and quantify the information leak. We show that Octopus provides near-optimal anonymity for both the lookup initiator and target. In a network of 100 000 nodes with 20% malicious nodes, Octopus only leaks 0.57 bit of information about the initiator and 0.82 bit of information about the target. We note that this is at least 4-6 times better than what previous works were able to achieve [28, 20]. As for efficiency, we evaluate Octopus on Planetlab with 207 nodes, and compare it with the base-line scheme Chord [34] and one of the state-of-art secure DHT lookups Halo [17]. The lookup latency of Octopus is comparable to Chord's, and even better than Halo's. While Octopus incurs relatively higher communication overhead than Chord and Halo to provide extra security and/or anonymity guarantees, the bandwidth consumption of Octopus is still manageable, which is only a few kbps for each node.

The remainder paper is organized as follows. We describe background material and related work in Section 2. Section 3 presents the system model, and Section 4 describes the design of Octopus. We analyze Octopus's security and anonymity in Section 5 and 6, and evaluate its efficiency in Section 7. Finally, we conclude in Section 8.

2 Background and Related Work

In DHT systems, each peer is assigned a unique ID associated with its IP address and/or public key, and maintains a short list (typically of size $\Theta(\log N)$, where N is the network size) of contact nodes (called *fingers*), whose IDs are determined by a mathematical formula designed to guarantee efficient routing. Each peer is responsible for a range of keys in the ID space. For example, in Chord [34], each node owns the keys from its predecessor's ID plus 1 to its own ID. A node can look up the owner of a given key either recursively or iteratively in $\Theta(\log N)$ hops. In a recursive lookup, the lookup initiator contacts one of its fingers, and that finger recursively passing the query along to one of fingers. In an iterative lookup, initiator contacts each intermediate node directly.

Secure DHT Lookups. Since the lookup initiator has to rely on other peers to locate the lookup target, malicious nodes could provide misinformation to bias the lookup result. A major school of proposals to securing DHT lookups uses *redundancy*. Castro et al. [7] propose a robust DHT system that relies on redundant lookups. Each key is replicated among several replica nodes

(typically the neighbors of the key owner). Instead of doing a single lookup, the initiator performs multiple redundant lookups towards all the replicas. The lookup result would be correct as long as one of the redundant lookup paths does not contain any malicious nodes. The limitation of this approach is that the redundant lookups tend to converge to a smaller number of nodes close to the target, and one malicious node in this set could infect many redundant lookups. Much subsequent work (such as [3, 27, 17]) therefore focuses on disentangling the redundant lookup paths to provide better security. Cyclone [3] proposes to partition nodes into r Chord sub-rings based on similarity of node IDs, and has r redundant lookups routed through the r sub-rings independently. Salsa [27] proposes a new virtual-tree-based DHT structure in which any two nodes share very few global contacts, so that redundant messages can proceed along different paths. Halo [17] tries to avoid changing the underlying DHT structures, but uses the original Chord overlay and performs redundant searches towards *knuckles* – nodes that have fingers pointing to the target.

While effective in ensuring security, these redundant-lookup-based approaches are incapable of preserving anonymity, since redundant messages create a lot of opportunities for an adversary to gain information about the lookup initiator and/or the target (refer to [22] for more details). ShadowWalker [23] proposes to embed redundancy into the DHT itself by building redundant topologies; instead of sending multiple redundant messages, the initiator uses *shadows* (nodes in redundant topologies) to verify each step of a lookup. Unfortunately, Schuchard et al. [31] find that ShadowWalker is vulnerable to *eclipse attack*, where an entire set of compromised shadows of a certain node could mount active attacks to rapidly infect routing states of other nodes in the network. They also show that increasing the dimension of redundant topologies could mitigate eclipse attack, but the resultant efficiency cost is prohibitively high.

Another major school of research on secure DHT lookups leverages cryptographic techniques. Mymric [37] uses an online certificate authority to generate certificates on nodes' routing states, thus limiting malicious nodes from providing bogus routing information. The major limitation of Myrmic is that for each node join/churn, the central authority has to update the certificates for all related nodes (e.g. all successors and predecessors of the new/churned node). Young et al. [26] propose two schemes RCP-I and RCP-II that use threshold signature and distributed key generation to mitigate the need of a central online authority. In their schemes, the verification information on each message is collaboratively generated by a threshold number of nodes, rather by a central authority.

All these secure DHT lookup schemes are not designed to preserve anonymity. Lookup keys are revealed during queries, and identities of lookup initiators are easily exposed due to directly contacting intermediate nodes.

Anonymous and Secure DHT Lookups. NISAN [28] is among the first to try to provide both security and anonymity guarantees in DHT systems. Its design is based on iterative Chord lookup. For security purpose, each queried node is required to provide its entire fingertable, so that the lookup initiator can apply *bound checking* on it to limit manipulation of fingertables. We note that bound checking is merely a moderate defense mechanism, as a malicious node could modify a few fingers without being detected. NISAN also uses redundancy to enhance security. They propose a greed-search mechanism to query multiple nodes at each step and combine the query results to tolerate misinformation. On the other hand, acquiring the entire fingertable also helps protect the anonymity of lookup targets, since lookup keys are not revealed to intermediate nodes. Nevertheless, NISAN can only provide very limited anonymity protection. Wang et al. [38] show that even though a lookup key is concealed, a passive adversary is still able to narrow the range of possibilities of a lookup target down to a small number of nodes, by analyzing the locations of observed queries (called *range estimation attack*). In addition, similar to vanilla and security-only DHT lookups, NISAN does not preserve initiator anonymity.

Torsk [20] is a DHT-based anonymous communication system. A key component of Torsk is a proxy-based anonymous DHT lookup. The idea is that a lookup initiator performs a random walk on the overlay to find a random node (called *buddy*), and requests the buddy to perform the lookup on its behalf; thus, the initiator does not need to contact intermediate nodes and its identity is protected. Because Torsk uses Myrmic [37] to secure lookups, it has the same limitation as Myrmic – requiring an online central authority to sign each node's routing state. In addition, as to be described in Section 4.2, a single proxy structure is insufficient to provide high levels of anonymity protection: information of lookup targets can be learnt by range estimation attack, and further be used to compromise the anonymous communication system built on top of the DHT lookup via *relay exhaustion attack* (refer to [38] for more details about this attack);

Recently, Backes et al. [4] propose to leverage *oblivious transfer* protocol to add query privacy to RCP-I and RCP-II [26]. However, for similar reasons as NISAN, this scheme is vulnerable to range estimation attack, and cannot protect initiator anonymity.

DHT-based Anonymous Communication Systems. One important application of anonymous and secure DHT lookups is to build scalable anonymous communication systems [27, 21, 20, 23]. In such systems, each node can conduct anonymous communication using a three-relay circuit as in Tor [13], and meanwhile can function as a relay to forward other nodes' traffic. Salsa [27] uses a customized DHT lookup to locate relays. AP3 [21] builds the circuit by performing a stochastic expected length random walk and uses Castro et al.'s DHT lookup [7] to find each hop on the random walk. Due to use of redundant lookups, both systems are vulnerable to information leak attacks [22]. Torsk [20] uses Myrmic [37] and secret buddy mechanism to locate relays. Due to lack of target anonymity in Myrmic, an adversary can launch selective Denial-of-Service attack [6] on Torsk, by exhausting buddies of the next hop during the circuit construction. ShadowWalker [23] does not rely on DHT lookups to build anonymous circuits; instead, the initiator performs a random walk over nodes' fingertables. As we mentioned before, unfortunately, ShadowWalker is found vulnerable to eclipse attack due to use of redundant topologies [31].

3 System Model

3.1 Design Goals

Pfitzmann and Hansen defined several relevant anonymity properties for message-based communication, such as sender and receiver anonymity [29]. We consider equivalent properties in the context of DHT lookups.

- *Initiator anonymity*: given a lookup target, it should not be possible to determine its initiator.
- *Target anonymity*: given a lookup initiator, it should not be possible to determine its target.
- *Query unlinkability*: given several queries with known targets, it should not be possible to find out if they came from the same initiator.

Most of existing secure DHT lookup schemes try to achieve *lookup correctness* and/or *fingertable correctness*, which are defined as follows.

- *Lookup correctness*: given a key, the initiator should be able to find the correct target.
- *Fingertable correctness:* each node should be able to maintain its fingers correctly.

Apart from these two security properties, an anonymous lookup should also achieve *fingertable trustiness*, which is defined as:

- *Fingertable trustiness*: any node should not be able to provide other nodes with an intentionally manipulated fingertable.

The reason is that an adversary's goal may not be to bias a lookup result, but to misdirect the lookup to query more malicious nodes so that more information can be gained about the initiator and/or target.

3.2 Threat Model

In the same vein of related works [21, 27, 28, 20, 23, 26, 4], we do not consider a global adversary that is capable of controlling the whole network and observing all communication traffic. Such a global adversary seems unpractical in large-scaled P2P networks. Instead, we assume a partial adversary that controls a fraction f of all nodes in the network (f is typically assumed to be up to 20%). Malicious nodes can behave in an arbitrarily malicious way, such as intercepting, modifying or dropping any messages going through them, or injecting fake messages to any other nodes . We also assume that malicious nodes can log any messages they have seen and access to a high-speed communication channel to share any information with very low transmission delay.

Also similar to related works, we do not attempt to solve the problem of Sybil attack [14] in this work. Defending sybil attack is an interesting research area that has drawn a lot of attentions; a number of effective solutions have been proposed, such as [5, 11, 40]. While these solutions are applicable to Octopus, we adopt a simple approach by having a central certificate authority (CA) issue certificates to nodes, to limit adversaries from arbitrarily creating fake identifies.

4 Octopus

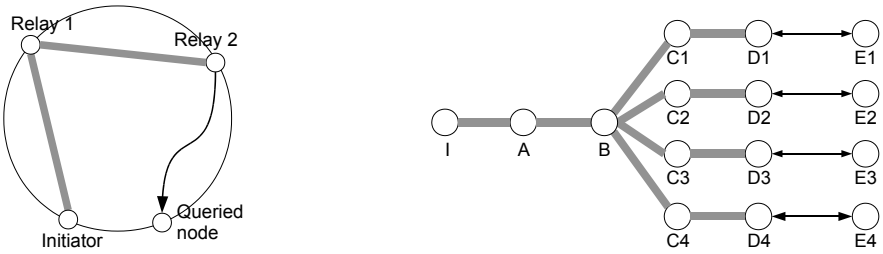
Now we describe the Octopus design. We motivate our design by studying fundamental challenges to building anonymous and secure DHT systems.

4.1 Challenge: Directly Exposing the Initiator and Target

There are two common anonymity issues with vanilla and security-only DHT systems. The first one is that lookup keys are simply revealed to intermediate nodes. In these schemes, a lookup initiator tells the value of its key to intermediate nodes, which then return their fingers closest to the key. Consequently, an adversary can learn the lookup target as long as one of the intermediate nodes is malicious. To address this, Octopus adopts a similar approach as NISAN by acquiring the entire fingertable from each intermediate node, thus hiding the lookup target. While this defense mechanism is not sufficient to achieve optimal target anonymity (due to range estimation attack [38]), information leak of lookup targets can be considerably reduced, compared with vanilla DHT lookups. Apart from this, Octopus also uses some advanced strategies to enhance anonymity, which we describe in Section 4.2.

The second issue is exposure of initiators' identities. In these systems, an initiator sends query messages to intermediate nodes directly; based on the source IP of received messages, intermediate nodes can easily infer the initiator's identity. If the adversary can associate a queried malicious node with the lookup target (e.g., based on the distance from the queried node to the target), she can link the initiator to its lookup target. We adopt a natural but powerful approach to addressing this problem. We let each initiator build an *anonymous path* using a few randomly selected nodes as *anonymization relays* to forward the lookup query/reply. As shown in Figure 1(a), the queried node can only see the IP of the last relay, and the source of the query is hidden behind the anonymous path; in addition, by using Onion encryption techniques [35], a single compromised relay is unable to know the identities of both the initiator and the queried node.

To ensure anonymity of the initiator, the relays must be independent from the initiator. We propose a specially designed random walk scheme to select relays. Using random walk for relay selection has been utilized in several anonymity systems. For example, Torsk [20] uses random walk to find buddy nodes, and ShadowWalker [23] uses it to construct anonymous communication



(a) Anonymous query through an anonymous path

(b) Structure of anonymous paths. (A, B), and (C_i, D_i) are pairs of anonymization relays, each of which are the last two hops of a random walk. E_i's are queried nodes.

Figure 1. Anonymous query and anonymous lookup in Octopus

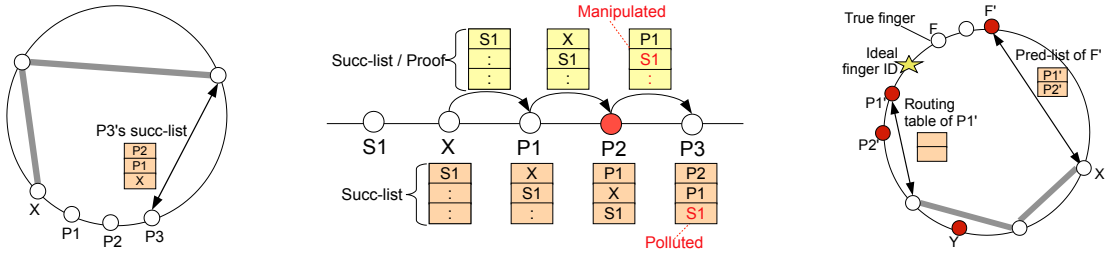
circuits. However, we note that securing random walk from active attacks is still a big challenge. A malicious node on the random walk could return misinformation (e.g. manipulated fingertable) to guide the initiator to choose a colluding node as the next hop, and consequently the initiator will end up choosing malicious nodes as relays. Torsk uses heavyweight cryptography to protect the random walk, which requires an online central authority to generate certificates for nodes' routing states. ShadowWalker relies on redundant topologies, which were found vulnerable to eclipse attack [31].

Octopus has two mechanisms to secure the random walk. One is a lightweight approach, which provides moderate protection: like NISAN, the initiator applies bound checking on the fingertables returned by intermediate nodes of the random walk to limit fingertable manipulation. The other is to proactively identify malicious nodes that manipulate fingertables and remove them from the network. Combination of the two mechanisms provides Octopus with strong security guarantee, without compromising the system scalability or introducing any new vulnerabilities. We elaborate the random walk scheme of Octopus in Appendix I and describe the detection algorithm for fingertable manipulation in Section 4.4.

4.2 Challenge: A Single Anonymous Path Is Insufficient

As we mentioned before, a single anonymous path is insufficient to achieve optimal target anonymity. Assuming that a common anonymous path is used for all queries in the lookup, an adversary can easily link observed queries (e.g., a query is observed by the adversary when the queried node is malicious) belonging to the same lookup, based on whether they are contacted by the same exit relay. Then using position information of observed queries, the adversary can apply range estimation attack [38] to reduce the candidate set of the target.

Octopus relies on two mechanisms to ensure target anonymity. First, we propose to use multiple anonymous paths in the lookup, with each query going through a separate path (the structure of anonymous paths/relays is shown in Figure 1(b)). This strategy can effectively disassociate the adversary's observations: the adversary only sees disjoint observation events from numerous concurrent lookups. Due to incapability of differentiating queries from different lookups, the adversary can gain much less information about each individual lookup. Second, we propose to add dummy queries in the lookup to further blur the adversary's observations and make range estimation attack even harder. We note that using multiple anonymous paths is very important to ensure effectiveness of dummy queries, because in the single-anonymous-path scenario, observed queries are linkable and thus dummy queries can be distinguished based on the positions of observed queries. In Section 6, we evaluate Octopus's anonymity with multiple anonymous paths and dummy queries taken into account.



(a) Secret neighbor surveillance

(b) Successor list pollution

(c) Secret finger surveillance

Figure 2. Malicious node identification

4.3 Challenge: Redundant Lookups Degrade Anonymity

As we mentioned before, using redundant lookups to limit lookup bias attack accelerates information leaks. Alternatively, we propose a novel security mechanism, called *secret neighbor surveillance*, which is performed independently from lookups and thus leaks no information about the lookup initiator and target. We motivate the design by studying the nature of lookup bias attack.

In original Chord DHT, a lookup ends when the closest finger (i.e. the direct successor) of a queried node succeeds the lookup key, i.e. the key is located between the queried node's ID and its successor's ID; then, the successor of the queried node is concerned as the key owner. Hence, a malicious intermediate node can bias the lookup result by reporting a colluding node that succeeds the key as its direct successor. Octopus is based on customized Chord DHT, and is subject to similar lookup bias attack. In particular, in Chord each node uses its fingertable for lookups and maintains a list of successors for stabilization, while in Octopus we let nodes also use their successor lists in lookups to speed up the lookup process in the last few hops, i.e. a combination of the fingertable and successor list (referred to as *routing table*) is returned by each intermediate node. Then, in Octopus a malicious intermediate node can launch lookup bias attack by manipulating its successor list in a way that the lookup key is between itself and one of its colluding "successors".

We observe that the key of thwarting lookup bias attack is to limit malicious nodes from manipulating successor lists. Therefore, we propose secret neighbor surveillance, a mechanism allowing nodes to monitor such misbehaviors of their neighbors. The basic idea is as follows. We let each node maintains a predecessor list, in the same way as maintaining the successor list (i.e., periodically running Chord stabilization protocol anti-clockwise). The predecessor list is of the same size as the successor list, and thus each node X should be contained in the successor list of any of its predecessors. As shown in Figure 2(a), from time to time, X sends an anonymous "lookup query" to a random predecessor (say P_3) through an anonymous path, and checks if itself is included in the returned successor list. Since P_3 cannot see the source of the query, it is unable to distinguish the testing query from real lookup queries. Thus, if P_3 tries to bias lookups (by removing honest nodes from its successor list), it will be detected by X, and X will report P_3 to the CA. To provide a non-repudiation proof on a manipulated successor list that is verifiable to the CA, each routing table is required to be signed and attached a time stamp by its owner.

Another strategy of lookup bias attack is to pollute honest nodes' successor lists during stabilization. For example, as shown in Figure 2(b), assuming P_2 is malicious and P_3 is honest, P_2 could send P_3 a manipulated successor list excluding X during P_3's stabilization, so that P_3 will concern X as dead and remove X from its successor list; consequently, P_3 will be identified by X by mistake and P_2 will still be uncovered. To deal with this, we let the successor list be signed by its owner and each node keep a queue of latest received successor lists as *proof*, to prove that its

successor list is not intentionally manipulated. For example, P_3 can provide its proof to the CA showing that its successor list is correctly computed according to the information provided by P_2. If P_3's proof is verified, then the suspicion on P_3 is cleared and the CA will request P_2 for its proof, and check it against P_3's proof. This process is repeated until finding a node that cannot provide valid proofs and this node will be judged as a malicious node.

4.4 Challenge: Fingertable Manipulation

Malicious nodes could manipulate not only successor lists, but also fingertables. For example, they could modify fingertables to misdirect a random walk, and also they could manipulate fingertables in a lookup to have more malicious nodes queried, creating more opportunities to gain information about the lookup.

We propose a novel defense mechanism, named *secret finger surveillance*, allowing nodes to secretly monitor integrity of others' fingertables. In particular, we let each node buffer a small number of received fingertables (e.g. from random walks, lookups, or secret security checks); as shown in Figure 2(c), from time to time, each node X chooses a random finger from one of the kept fingertables, say the i-th finger F' of node Y, and asks F' for its predecessor list. After a short random period of time, X sends an anonymous "lookup query" to a random predecessor of F' (say P'_1) through an anonymous path, and checks if the true finger exists in P'_1's successor list (i.e. if any node is closer to the ideal finger ID than F').

The intuition behind this is that if Y replaces a honest finger F with a malicious node F', then at least one of F''s (true) predecessors should be closer to the ideal finger ID than F'. Hence, if F' provides X with its true predecessor list, the fingertable manipulation will be detected. Therefore, F' has to manipulate its predecessor list, so that all the "predecessors" are malicious nodes and precede the ideal finger ID. On the other hand, we note that P'_1 cannot freely manipulate its successor list to meet the above requirements, because P'_1 is monitored by secret neighbor surveillance. Therefore, if the adversary tries to manipulate a single finger ($F \rightarrow F'$), she has to sacrifice at least one malicious node, either P'_1 or F' and Y.

4.5 Challenge: Fingertable Pollution and Secure Finger Update

Another major challenge in DHT systems is secure finger update. Similarly to Chord, Octopus lets each node periodically perform (non-anonymous) lookups towards ideal finger IDs. Such lookups are subject to lookup bias attack, and the goal of adversaries is to introduce more malicious nodes in honest nodes' fingertables (referred to as *fingertable pollution attack*). We can use a similar mechanism as secret finger surveillance to limit finger pollution attack. In particular, when X obtains the result (say F') of the finger-update lookup, it asks F' for its predecessor list, and chooses a random predecessor P'_1 of F' to perform the same checks as in secret finger surveillance. X uses F' to update its fingertable only when F' passes all security checks.

4.6 The CA and Certificate Revocation

In Octopus, the CA is in change of certificate management, including issuing new certificates and revoking certificates from identified malicious nodes. We note that our CA needs to be online only for a short period of time with very limited workload. As we shall show in Section 5, Octopus is very effective in identifying malicious nodes: all attacking nodes can be identified in a short time (e.g. 30 mins), and after that, there will be no new workload for the CA. We also show that (in Section 7) even during the busiest time, the workload for the CA is still manageable (only processing a few messages per second). It is important to note that the CA of Octopus is fundamentally different from that of Myrmic [37] and Torsk [20]. The latter is required to be online all the time and needs to update certificates for multiple nodes for each node churn/join. Whereas, the certificates in Octopus are in nature the same as common identity certificate (like X.509 [1]);

Table 1. Error rate of end-to-end timing analysis attack. α is concurrent lookup rate.

Max. delay	$\alpha = 0.5\%$	$\alpha = 1\%$	$\alpha = 5\%$
100 ms	99.35%	99.50%	99.91%
200 ms	99.60%	99.82%	99.95%

they are independent with nodes' routing states and thus do not need to be updated frequently.

In Octopus, malicious nodes are removed from the network via certificate revocation. Revoking certificates in Public Key Infrastructure (PKI) has been extensively studied in the past twenty years. Several fairly efficient and scalable revocation mechanisms have been proposed, such as Merkle Hash Tree based certificate revocation [25], efficient distribution of revocation information over P2P networks [24], and scalable PKI based on P2P systems [18]. Since certificate management in Octopus has no essential difference from those of traditional PKIs, these mechanisms are directly applicable to Octopus; yet, certificate management is not the focus of this paper.

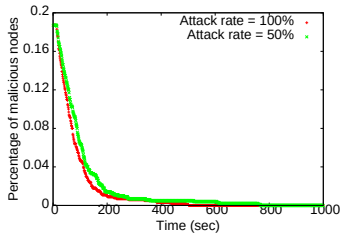
4.7 Other Attacks, Countermeasures, and Discussion

End-to-End Timing Analysis Attack. End-to-end timing analysis attack is an attack that associates malicious relays on the same anonymous path (e.g. A and D_i in Figure 1(b)), by analyzing timings of packets in the traffic going through them. Since in Octopus there is only one message transmitted through each anonymous path in either forward or backward direction, timing analysis attacks that require a large number of observed packets (such as packet counting [32] or packet timing correlation [39, 10]) is inapplicable to Octopus.

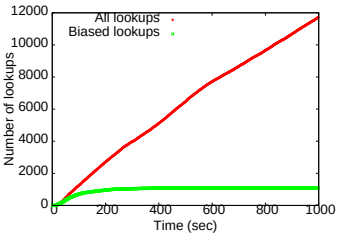
For Octopus, adversaries could try to link A and D_i based on similarity of upstream and downstream latencies: in a noise-free network environment, the transmission latency from A to D_i should be the same as that from D_i to A. However, this similarity depends on communication latency jitters and can be easily destroyed by adding a short random delay at a middle relay. We simulate this attack using King dataset [9] to find a pair of A and D_i with the smallest difference between upstream and downstream latencies. The delay is introduced at relay B and chosen randomly between 0 and a pre-set maximum delay. We choose a typical network setting as used in related works [38, 22, 23, 28]: $N = 1\,000\,000$ nodes, 20% malicious nodes, and concurrent lookup rate α between 0.5% - 5%. We set the jitter window as 10 ms or 10% of transmission latency whichever is smaller (according to [2]). Table 1 presents the simulation results. When the maximum delay is 100 ms and $\alpha = 5\%$, the error rate is as high as 99.91%; in this case the information leak is only $(1 - 99.91\%) \cdot \log_2(N \cdot 0.8 + N \cdot \alpha \cdot 0.2) = 0.018$ bit, which means adversaries can hardly learn any information from timing analysis attack.

Selective Denial-of-Service Attack A threat to anonymous communication systems (like Tor [13]), selective Denial of Service (DoS) attack [6], can increase the chance of compromising anonymous circuits by selectively dropping packets to tear down the circuits that are infeasible to compromise. Selective DoS attack is also applicable to Octopus. For example, to create more opportunities of observing initiators, malicious relays could selectively drop lookup queries or replies, when the relay directly connected to the initiator is not malicious. Nevertheless, under Octopus's attacker identification framework, selective DoS attack can be effectively constrained by identifying malicious droppers. We describe and evaluate our selective-DoS-attack defense in Appendix II.

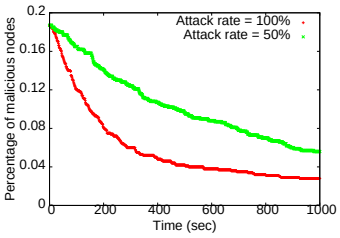
Relay Exhaustion Attack. Relay exhaustion attack [38] is a selective-DoS-flavored attack, used to compromise DHT-based anonymity systems that are lack of target anonymity protection. In such an attack, adversaries can utilize information leak of the lookup target to predict the next hop of the circuit construction and launch flooding-based attack to prevent the circuit from being extended to the next hop. We note that Octopus is resistant to relay exhaustion attack, since little information about the target is leaked in Octopus. We present the formal anonymity analysis in



(a) Lookup bias attack



(b) Number of biased lookups



(c) Fingertable manipulation attack

Figure 3.

Section 6.

5 Security Evaluation

In this section, we experimentially evaluate our malicious node identification mechanisms in limiting active attacks, including lookup bias attack, fingertable manipulation attack, and fingertable pollution attack. We use four evaluation metrics as follows.

- fraction of remaining malicious nodes in the network as time goes by
- false positive rate, i.e. the chance that a honest node is judged as malicious
- false negative rate, i.e. the chance that a malicious node is not identified when being tested by a node
- false alarm rate, i.e. the chance that there is no node (either honest or malicious) identified in a report sent to the CA

These metrics represent different aspects of security properties. Reduction of fraction of malicious nodes shows *effectiveness*, false positive/negative rates represent *accuracy*, and false alarm rate indicates *efficiency*.

5.1 Experiment Setup

We developed an event-based simulator for the security evaluation. The simulator is written in C++ with about 3.0 KLOC. We consider a WAN setting, where latencies between each pair of peers are estimated using the King dataset[2] [9]. We model node churn/join as an exponential distribution process $f(x) = \lambda e^{-(1/\lambda)x}$ with mean life time λ minutes. We generate random network topologies of size $N = 1000$ with 20% malicious nodes. Each node maintains 12 fingers and 6 successors/predecessors. To handle churn, each node runs successor and predecessor stabilization protocols every 2 seconds, and performs lookups for finger updates every 30 seconds. To discover malicious nodes, each peer performs security checks of secret neighbor surveillance and secret finger surveillance every 60 seconds, and performs a random walk for relay selection every 15 seconds. To ensure high accuracy in malicious node identification, each node keeps 6 latest received successor lists as proofs. For lookup-related evaluation, we let each node perform one lookup every minute.

5.2 Experimental Results

Lookup Bias Attack. Figure 3(a) shows the malicious node reduction in lookup bias attack, and Figure 3(b) shows the number of biased lookups. We see that the secret neighbor surveillance mechanism can rapidly identify malicious nodes and effectively limit lookup bias attack. After a short time (20 mins), almost all malicious nodes are discovered, and there is no new lookup being

[2]King dataset contains measured latencies between Internet domain name servers (DNS) and is highly heterogeneous. The average round trip time (RTT) is around 182 ms.

Table 2. False positive/negative/alarm rates of malicious node identification mechanisms. λ is mean life time of each node (in minute). Attack rate is 100%. In fingertable manipulation/pollution attacks, checked malicious predecessors provide manipulated successor lists with 50%.

Attacks	False Positive		False Negative		False Alarm	
	$\lambda = 60m$	$\lambda = 10m$	$\lambda = 60m$	$\lambda = 10m$	$\lambda = 60m$	$\lambda = 10m$
Lookup Bias	0	0	0	0.52%	0	0.52%
Fingertable Manipulation	0	0	14.02%	19.55%	0.18%	1.55%
Fingertable Pollution	0	0	14.08%	18.48%	0.33%	2.18%

biased. In addition, we note that the more aggressive malicious nodes are, the faster they will be caught.

The accuracy properties of our attacker identification mechanisms are shown in Table 2. We note that Octopus has 0 false positive rate for any of the identification mechanisms, even when the churn rate is very high (e.g. the mean life time for each node is 10 mins). This ensures that honest nodes will not be judged as malicious nodes by mistake. The high accuracy of secret neighbor surveillance mechanism is also shown from the low false negative, which is less than 0.6% under frequent churns.

We note that it is possible that nodes churn during the investigation process, although we did not observe any such occurrences in our experiments even when the churn rate is very high. For example, node X could churn right after replying to a lookup query, and becomes unresponsive when being requested by the CA to provide a proof. In such cases, the CA cannot tell whether X is really dead or pretends to be dead to avoid being identified. To deal with this, we could let the CA judge a "churned" node who has churned once in recent investigations (e.g. in 12 hours) as a malicious node.

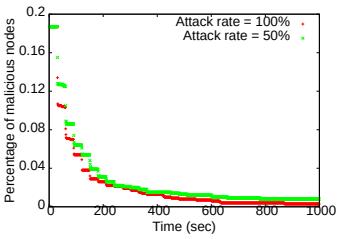


Figure 4. Fingertable pollution

Fingertable Manipulation/Pollution Attack. Fingertable manipulation attack aim to bias random walks or create more opportunities for an adversary to gain information about a lookup; fingertable pollution attack attempts to achieve these goals by corrupting honest nodes' fingertables. We can see from Table 2 that the identification mechanisms for both attacks have higher false negative rates than secure neighbor surveillance. The reason is that a malicious finger could pass the security checks, if the randomly selected predecessor for consistence checking happens to be a colluding node and provides a successor list consistent with the malicious finger. However, Figure 3(c) and Figure 4 show that both identification mechanisms are still very effective in discovering malicious nodes: over 80% malicious nodes can be identified within 30 mins.

Furthermore, we notice that the identification mechanism for fingertable pollution attack can identify malicious nodes relatively faster than secure finger surveillance. This is because 1) the former is applied more frequently (at each finger update) and 2) some fingers that are contained in the successor list are also monitored by secret neighbor surveillance mechanism.

6 Anonymity Analysis

In this section, we analyze the best strategies for an adversary to infer the lookup target T and the initiator I based her observations, and calculate the target anonymity $H(T)$ and the initiator anonymity $H(I)$, respectively. We use entropy – a standard information theoretic metric for anonymity evaluation [15] – to quantify $H(T)$ and $H(I)$. We let O denote the set of possible observations gained by the adversary (including null observation). To measure the system as a whole, we have:

$$H(T) = \sum_{o \in O} P(o) \cdot H(T|o) \qquad H(I) = \sum_{o \in O} P(o) \cdot H(I|o) \tag{1}$$

where $P(o)$ is the probability of observation o occurring.

To calculate the maximum information leak, we make the following assumptions in anonymity calculations. First, we assume the network is static, since network dynamics can obscure the adversary's observations and make extracting information about the initiator/target even harder. Second, we assume that active attacks are not mounted, as malicious nodes will be quickly identified by the identification mechanisms and the adversary will lose observation spots for passive attacks.

6.1 The Adversary's Observations

An observation o consists of a (large) number of observed events, which are message transmissions seen by malicious nodes. Each observed event can provide information, such as sender/receiver IDs, message content, and transmission time. The adversary can log all the observed events, and try to drive useful information from any combinations of them.

Observations of queries. There are two cases where a query is observed (i.e. the queried node is identified): 1) the queried node itself is malicious, or 2) the exit relay is malicious. The adversary can link an observed query backwards to I in two ways. One is through direct connection of compromised relays on the anonymous path. For example, if the relays A and C_i and the queried node E_i are malicious (refer to Figure 1(b)), E_i is observed and can be linked to I using C_i and A as bridges. The other is through linkability of relays to I in the random walk. For instance, if D_i is compromised and linkable to I in the random walk, then E_i is observed and linkable to I using D_i as a shortcut. It is possible to use both approaches at the same time.

Furthermore, considering all queries in the same lookup together can help the adversary link more queries to I. Since all (C_i, D_i) in the same lookup are connected to the same relay B, if there exists one query linkable to both I and B, then any other queries that are linkable to B can also be linked to I. In the rest paper, we use "a linkable query" to specifically refer to a query that is *observed* and *linkable* to I.

Observations of the initiator. The adversary's goal is to link I and T; knowing only one of them is useless to the adversary. Therefore, the pre-condition of compromising target anonymity is to observe I. Since I is directly connected to A, the adversary can observe I as long as A is malicious. In addition, I is also observable in random walks. Thus, another case for I being observed is that there exists at least one malicious relay linkable to I in a random walk.

Observation of the target. For similar reasons, in order to compromise initiator anonymity, the adversary has to know T. We note that T is not necessarily contacted during the lookup, since the aim of the lookup is to find the IP address of T, which will be learnt from query replies of intermediate nodes (after the lookup is done, T might be contacted due to some application needs, but we do not consider this as part of the lookup). Yet we assume that each node can tell whether itself is a target node based on its role in the application. For example, in DHT-based anonymous communication, a node can learn that it is a lookup target if it is selected as a relay of an anonymous circuit. Therefore, we concern T as observed when T itself is a malicious node.

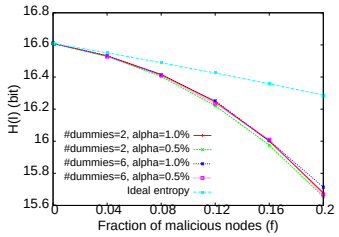
6.2 Initiator Anonymity

To calculate the initiator anonymity, we divide the observations of the adversary into two categories.

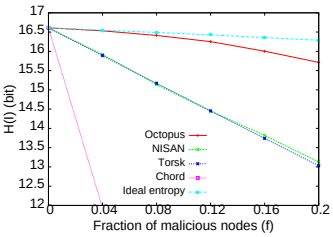
- o_n: the observation occurring when T is not observed
- O_o: the set of observations occurring when T is observed

According to 1, $H(I)$ is calculated as:

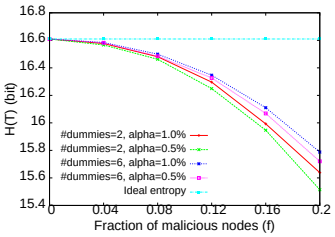
$$H(I) = P(o_n) \cdot H(I|o_n) + \sum_{o_o \in O_o} P(o_o) \cdot H(I|o_o) \tag{2}$$



(a) Initiator anonymity of Octopus. α is concurrent lookup rate



(b) Comparison of initiator anonymity: $\alpha = 1\%$



(c) Target anonymity of Octopus

Figure 5.

When T is not observed, the entropy of I is maximized. Presuming that the adversary can exclude malicious nodes from the anonymity set of I, we have:

$$H(I|o_n) = \log_2((1-f)\cdot N) \tag{3}$$

Let $\mathcal{R}_T^l$ denote the set of non-dummy queries linkable to I in the lookup whose target is T. Then, based on whether $\mathcal{R}_T^l$ is an empty set, we calculate $H(I|o_o)$ as follows:

$$H(I|o_o) = P(\mathcal{R}_T^l = \emptyset)\cdot H'(I|o_o) + (1 - P(\mathcal{R}_T^l = \emptyset))\cdot H''(I|o_o) \tag{4}$$

When there is no linkable non-dummy query, I is unlinkable with T. However, since some of the initiators of concurrent lookups can be observed by the adversary, we have:

$$H'(I|o_o) = P(I\_obsv)\cdot\log_2(\#obsv\_hon\_init) + (1 - P(I\_obsv))\cdot\log_2((1-f)\cdot N) \tag{5}$$

Let Ψ^l denote the set of concurrent lookups that have at least one linkable query, and ψ_T denote the lookup with target T. When $\mathcal{R}_T^l \neq \emptyset$, $\psi_T \in \Psi^l$. Each lookup in Ψ^l is possible to be ψ_T. Therefore, we have:

$$H''(I|o_o) = -\sum_{\psi\in\Psi^l} P(\psi = \psi_T|o_o)\cdot\log_2 P(\psi = \psi_T|o_o) \tag{6}$$

Because the density of queries close to the target is higher than other regions on the ring, for ψ_T it is highly likely that the last queried node in $\mathcal{R}_T^l$ is located very close to T. Therefore, the adversary can assign probability to each candidate initiator based on the minimum distance (i.e. number of hops) between its queried nodes and T. In particular, let $\mathcal{Q}_\psi^l$ denote the set of linkable queries in ψ, $\psi \in \Psi^l$, and let $\xi(x)$ denote the probability that for ψ_T the minimum distance from linkable queried nodes to T is x. $\xi(x)$ can be obtained via pre-simulations of the lookup. Then, we can calculate $P(\psi = \psi_T|o_o)$ as follows:

$$P(\psi = \psi_T|o_o) \approx \frac{\xi(\min_{E\in\mathcal{Q}_\psi^l} dist(E,T))}{\sum_{\psi'\in\Psi^l}\xi(\min_{E'\in\mathcal{Q}_{\psi'}^l} dist(E',T))} \tag{7}$$

We develop a simulator for initiator anonymity measurement in C++ with about 0.8 KLOC. The results are shown in Figure 5(a). With network size $N = 100\,000$, concurrent lookup rate $\alpha = 1\%$, $f = 20\%$ malicious nodes, and 6 dummies, Octopus only leaks 0.57 bits of information about the initiator. We also calculate the initiator anonymity of the base-line scheme Chord [34] and the state-of-the-art anonymous and secure DHT lookups NISAN [28] and Torsk [20]. The comparison is provided in Figure 5(b). We can see that in the same setting, NISAN and Torsk leak about 3.3 bits of information, which is about 6 times more than Octopus. Here we do not explicitly compare Octopus with redundant-lookup-based DHTs (such as Halo [17]), since they leak more information than the base-line system Chord.

We also note that adding more dummy queries does not really help improve initiator anonymity. This is because dummy queries are introduced mainly to blur the adversary's observations on the target to improve target anonymity.

Table 3. Efficiency comparison. LK_INT is the time interval between two consecutive lookups.

Schemes	Lookup Latency (sec)		Bandwidth Consumption (kbps)	
	Mean	Median	$LK\_INT = 5min$	$LK\_INT = 10min$
Octopus	2.15	1.61	5.91	4.30
Chord [34]	1.35	0.35	0.29	0.28
Halo [17]	6.89	1.79	0.71	0.37

6.3 Target Anonymity

The calculation of target anonymity is more complex than that of initiator anonymity. Due to space limitation, we provide the detailed calculation of target anonymity in Appendix III. We develop a simulator for target anonymity measurement in C++ with about 1.3 KLOC. When $f = 20\%$ and #dummies $= 6$, Octopus leaks 0.82 bits of information about T (see Figure 5(c)). The anonymity grows with more added dummy queries. In this same setting, the information leak for NISAN land Torsk is 11.3 bits and 3.4 bits, respectively (see Figure 6). We note that our calculation of $H(T)$ is based on the preconditions that I is observed (i.e. the adversary should link T to I). The secret buddy mechanism of Torsk is effective in unlinking I from T, but is unable to protect the anonymity of T itself (i.e. the adversary can easily learn T although the corresponding I may be unknown). This can make the application built on top of the DHT vulnerable to relay exhaustion attack [38]. By contrast, for Octopus, due to the design of multiple anonymous paths and dummy queries, adversaries can learn little information about T no matter I is observed or not; thus, Octopus is robust to relay exhaustion attack.

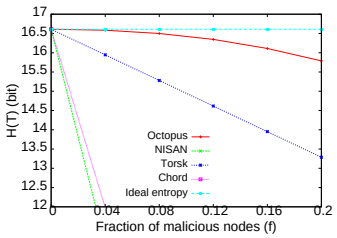


Figure 6. Comparison of target anonymity: $\alpha = 1\%$

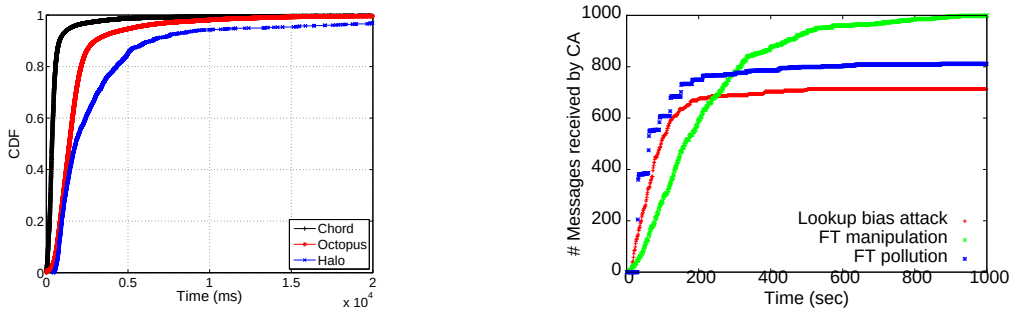
7 Efficiency Evaluation

Lookup Latency. Lookup latency is one of the most important performance factors for DHT systems. We measure the lookup latency of Octopus using PlanetLab with 207 randomly selected nodes. We use boost C++ library[3] (mainly UDP asynchronous read/write of Boost.Asio) to build the communication substrate. We let each node perform 2000 lookups independently using randomly picked lookup keys. For each lookup, we record the latency from the time of sending out the first query till the time of receiving the lookup result. We let the second relay B add a random delay up to 100 ms on forwarded messages to limit timing analysis attack.

For comparison, we use the same methodology to implement Chord [34] and Halo [17], and measure their lookup latencies in the same network environment. We choose Halo because it is one of the state-of-the-art secure DHT lookup schemes and it is also based on Chord overlay. For Halo, we use degree-2 recursion with redundant parameter 8×4, as suggested in their paper [17] to provide fairly strong security guarantee. The experimental results are presented in Figure 7(a) and Table 3. We can see that while the lookup latency of Octopus is relatively longer than Chord's due to more communication for security and anonymity needs, it is shorter than Halo's, for which only security guarantee can be provided. The outperformance is because Octopus does not rely on redundant lookups; in Halo, a lookup is not completed until all redundant lookups' results are returned.

Bandwidth Overhead. We also compare Octopus with Chord and Halo in terms of bandwidth cost. We adopt the same configuration as described in Section 5 for each of the DHT lookups, and

[3] www.boost.org



(a) CDF of lookup latency

(b) The CA's workload

Figure 7. Efficiency evaluation

consider an overlay network with 1 000 000 nodes[4]. We can see from Table 3 that Octopus does incur higher communication overhead than Chord and Halo, in order to achieve high levels of anonymity protection; however, the bandwidth cost of Octopus is still reasonable (only a few kbps), which is affordable even for low-end clients with limited connection bandwidth.

The CA's Workload. Octopus uses the CA to process attack reports and identify malicious nodes. We use the event-based simulator (described in Section 5) to measure the workload of the CA in terms of the number of messages (including reports, proofs, and etc) received by the CA. We can see from Figure 7(b) that the peak of the CA's workload occurs at the beginning phase of the network deployment, at which time the number of malicious nodes is maximal. As time goes by, there are fewer malicious nodes remaining in the network, and thus fewer reports need to be processed by the CA. After 20 min, there is hardly any new reports for the CA. Even during the busiest time (the first 10 min), the CA only needs to process about 2 messages per second on average, which can be handled by most Internet servers.

8 Conclusion

In this paper, we present Octopus, a novel anonymous and secure DHT lookup, which provides strong guarantees for both anonymity and security. Octopus ensures anonymity and security via three fundamental techniques. First, Octopus constructs an anonymous path to relay lookup query messages while hiding their initiator. Second, it splits the individual queries used in a lookup over multiple paths, and introduces dummy queries, to make it difficult for an adversary to learn the lookup target. Third, it uses secret security checks to identify and remove misbehaving malicious nodes. We systematically evaluate the security and anonymity of Octopus. With the developed event-based simulator, we show that malicious nodes can be quickly identified with high accuracy. In addition, via probabilistic modeling and simulation, we show that Octopus can achieve near-optimal anonymity for both the lookup initiator and target. It only leaks 0.57 bits of information about the initiator and 0.82 bit of information about the target in a network of 100 000 nodes and 20% malicious nodes. This is 4-6 times better than what previous work was able to achieve. We also evaluate the efficiency of Octopus on Planetlab with 207 nodes, and show that Octopus has reasonable lookup latency and communication overhead.

[4]We use the following parameters to estimate the bandwidth overhead. Each routing state item (such as fingers or successors) is 10 bytes. We use ECDSA signature (40 bytes) for authentication with a 4-byte timestamp, and AES-128 for onion encryption. Each certificate is 50 bytes, including the node's IP address (6 bytes), the node's public key (20 bytes), expire time (4 bytes), and the CA's signature (20 bytes).

References

[1] Itu/iso recommendation (2000) internet x.509 information technology open systems interconnection the directory: authentication frameworks, technical corrigendum.

[2] A. Acharya and J. Saltz. A study of internet round-trip delay. *Technical Report (CS-TR 3738), University of Maryland*, 1996.

[3] M. S. Artigas, P. G. Lopez, J. P. Ahullo, and A. F. G. Skarmeta. Cyclone: A novel design schema for hierarchical dhts. *In P2P 05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 49–56, 2005.

[4] M. Backes, I.Goldberg, A. Kate, and T. Toft. Adding query privacy to robust dhts. *Tech. rep., arXiv:1107.1072v1 [cs.CR]*, July 2011.

[5] N. Borisov. Computational puzzles as sybil defenses. In *Peer-to-Peer Computing*, 2006.

[6] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? *ACM CCS*, 2007.

[7] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI*, December 2002.

[8] L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. *Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, 2009.

[9] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a dht for low latency and high throughput. *NSDI*, 2004.

[10] G. Danezis. The traffic analysis of continuous-time mixes. In *Privacy Enhancing Technologies workshop*, May 2004.

[11] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[12] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase mix-net reliability. In *the 4th International Workshop on Information Hiding (IHW)*, 2001.

[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, August 2004.

[14] J. Douceur. The sybil attack. In *Peer-to-Peer Systems*, volume 2429, pages 251–260. 2002.

[15] C. Daz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 184–188. Springer Berlin / Heidelberg, 2003.

[16] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *Proc. of NSDI'04*, pages 18–18, Berkeley, CA, USA, 2004. USENIX Association.

[17] A. Kapadia and N. Triandopoulos. Halo: High-assurance locate for distributed hash tables. In *NDSS*, February 2008.

[18] C. Y. Liau, S. Bressan, K.-L. Tan, C. Yee, L. Stphane, and B. K. lee Tan. Efficient certificate revocation: A p2p approach. In *HICSS'05*, 2005.

[19] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *IPTPS*, 2001.

[20] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. *ACM CCS*, November 2009.

[21] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. Ap3: Cooperative, decentrialized anonymous communication. *ACM SIGOPS European Workshop*, 2004.

[22] P. Mittal and N. Borisov. Information leaks in structured peer-to-peer anonymous communication systems. *ACM CCS*, 2008.

[23] P. Mittal and N. Borisov. Shadowwalker: Peer-to-peer anonymous communication using redundant structured topologies. *ACM CCS*, November 2009.

[24] M. C. Morogan and S. Muftic. Certificate revocation system based on peer-to-peer crl distribution. In *Proc. of the DMS'03 Conference*, 2003.

[25] J. L. Muoz, J. Forne, O. Esparza, and M. Soriano. Certificate revocation system implementation based on the merkle hash tree. *Inte. Journ. of Inf. Sec.*, 2(2), 2003.

[26] M.Young, A.Kate, I.Goldberg, and M.Karsten. Practical robust communication in dhts tolerating a byzantine adversary. *Proc. ICDCS'10*, pages 263–272, 2010.

[27] A. Nambiar and M. Wright. Salsa: A structured approach to large-scale anonymity. *ACM CCS*, 2006.

[28] A. Panchenko, S. Richter, and A. Rache. Nisan: Network information service for anonymization networks. *ACM CCS*, November 2009.

[29] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, Aug. 2010. v0.34.

[30] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proc. SOSP'01*, pages 188–201. ACM, 2001.

[31] M. Schuchard, A. Dean, V. Heorhiadi, N. Hopper, and Y. Kim. Balancing the shadows. *ACM WPES*, 2010.

[32] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *ESORICS'03*, October 2003.

[33] A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres. Privacy, cost, and availability tradeoffs in decentralized osns. WOSN '09, pages 13–18, 2009.

[34] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[35] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. *International Workshop on Design Issues in Anonymity and Unobservaility, vol. 2009, LNCS, Springer*, July 2000.

[36] D. Wallach. A survey of peer-to-peer security issues. In *Software Security Theories and Systems*, volume 2609 of *LNCS*, pages 253–258. 2003.

[37] P. Wang, I. Osipkov, N. Hopper, and Y. Kim. Myrmic: Secure and robust dht routing. *Tech. rep., Digital Technology Center, University of Minnesota at Twin Cities*, 2007.

[38] Q. Wang, P. Mittal, and N. Borisov. In search of an anonymous and secure lookup: Attacks on structured peer-to-peer anonymous communication systems. *ACM CCS*, 2010.

[39] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *IEEE Security and Privacy*, May 2007.

[40] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy (Oakland)*, 2008.

Appendix I: Random Walk for Relay Selection

As shown in Figure 8, the random walk originates from the initiator I and is composed of two phases, with l nodes visited in each phase. The motivation of dividing the random walk into two phases is to mitigate timing analysis attack. In the first phase, I picks a random finger U_1 out of its fingertable, and requests U_1 for its fingertable, from which the second hop U_2 is selected. Then I sends an onion-encrypted query to U_2, using U_1 as the forwarding node, and selects the third hop U_3 at random from the fingertable returned by U_2. This process is recursively repeated for l hops. To provide integrity check and source authentication, each replied fingertable is signed by its owner with the owner's certificate attached.

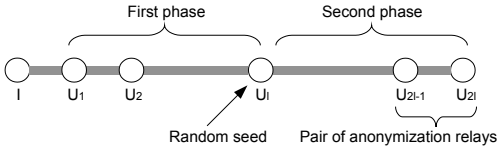


Figure 8. Two-phase random walk for anonymization relay selection.

The second phase of the random walk is conducted by U_l, the last visited node in the first phase. In particular, I sends U_l a random seed through the anonymous path established in the first phase, and the seed will guide U_l how to pick nodes "randomly"[5]. The second phase is performed in the same way as the

[5]For example, we could let U_l apply hash function to the seed for i times and map the hash value to $[1, m]$ (m is the size of fingertables) and use the result as an index to select the i-th hop.

first phase, and the last two hops (U_{2l-1} and U_{2l}) are chosen as a pair of anonymization relays. To prevent malicious U_l from biasing the random walk, U_l is required to keep all received fingertables, signatures and certificates, and send them back to I through the anonymous path of the first phase at the end of the random walk. Such information allows I to verify whether U_l has honestly performed the random walk. If the verification is invalid or I does not receive the results by a pre-set deadline, I chooses another node from U_{l-1}'s fingertable to restart the second phase of random walk.

Appendix II: Defense to Selective DoS Attack

To thwart selective DoS attack in Octopus, we leverage reputation-based reliability enhancement strategy for mix networks [12] to identify malicious dropper nodes. The idea is as follows. Each message is assigned a deadline by which it must be sent to the next hop (in either direction along the anonymous path). A relay X first tries to send the message to the next hop Y directly. If Y is alive and honest, it will send a signed *receipt* back to X. If X has not received a receipt from Y by a specified period before the deadline, it will request a pre-defined set of *witnesses* (e.g. its successors and predecessors) to independently try to send the message to Y and obtain a receipt. If a witness gets a valid receipt, it will forward it to X; otherwise, it sends X a signed *statement* to the delivery failure. Before sending any query, the initiator first checks if the first pair of relays A, B are alive (B is pinged through A). During the lookup, if the initiator does not receive the i-th query reply by the pre-set deadline, it queries the successors and predecessors of the relays C_i, D_i (through the partial anonymous path A and B) about their aliveness, which can be inferred based on their recent stabilization activities. If both of them are alive, the initiator reports the failure to the CA with the identities of all the relays. Then the CA will request the relays to provide either receipts or statements, and based on the provided information, the CA will be able to identify the malicious dropper node.

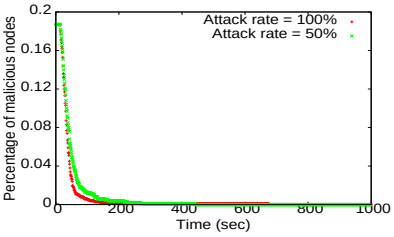


Figure 9. Selective DoS attack

DoS attacks are also possible in random walk. For example, a malicious hop of the random walk could simply drop packets to prevent the random walk from being completed, or a malicious U_l could deny returning the random walk result to the initiator if the result only contains honest nodes. We can use the same strategy as above to identify such malicious nodes. We use the event-based simulator (described in Section 5) to evaluate this defense mechanism. The simulation results are shown in Figure 9.

Appendix II: Calculation of Target Anonymity

We categorize the adversary's observations into three classes:

- o_n: the observation occurring when I is not observed
- O_l: the set of observations occurring when there is at least one linkable query in the lookup
- O_d: the set of observations occurring when there is no linkable query in the lookup

According to Equation (1), we have:

$$H(T) = P(o_n) \cdot H(T|o_n) + \sum_{o_l \in O_l} P(o_l) \cdot H(T|o_l) + \sum_{o_d \in O_d} P(o_d) \cdot H(T|o_d) \tag{8}$$

Since the adversary has to know I at the first place, the entropy of T is maximum when I is not observed, i.e. $H(T|o_n) = \log_2 N$.

Calculation of $H(T|o_l)$. When there exist queries that are linkable to I, the adversary can adopt range estimation attack to narrow the range of T. The output of this attack is a lower bound and an upper bound of T's location on the ring.

We temporarily assume that all queries observed by the adversary are non-dummy queries (how to deal with dummy queries is shown later). Suppose there are two or more linkable queries in the lookup. Let E_i and E_j denote the first and the last linkable queried nodes, respectively. Since nodes succeeding T will not be queried in the lookup, E_j can be used as a lower bound of T. An upper bound of T can be obtained based on the fact that the lookup always greedily queries the finger that is precedingly closest to the target. In particular, the adversary first decides the queried nodes between E_i and E_j by (locally) simulating the lookup from E_i to E_j. The initial upper bound is set as E_i; then for each pair of consecutive queries (E_k, E_{k+1}) between E_i and E_j, $i \leq k \leq j-1$, the adversary finds out the index of E_{k+1} in E_k's finger table (say p) and uses the $(p+1)$-th finger of E_k to update the upper bound.

Since the density of queries close to T on the ring is higher than other regions, nodes located closer to T in the estimation range are more likely to be T. We let $\gamma(i, z)$ denote the probability that the i-th node (clockwise) in an estimation range of size z is the target, $1 \leq i \leq z$. The probability distribution of $\gamma(i, z)$ can be obtained by pre-simulation of the lookup. Note that the range estimation attack is inapplicable when the lookup has only one linkable query (say E_i), but the adversary can use the successor of E_i as the lower bound of T and the predecessor of E_i as the upper bound of T, and assign higher probabilities to the nodes closer to the lower bound in the estimation range.

Now we discuss how to deal with dummy queries. Let $\mathcal{Q}_I^l$ denote the set of linkable queries in the lookup performed by I, and $\mathcal{R}_I^l$ denote the set of linkable non-dummy queries, $\mathcal{R}_I^l \subseteq \mathcal{Q}_I^l$. Based on whether $\mathcal{R}_I^l$ is an empty set, $H(T|o_l)$ can be calculated as:

$$H(T|o_l) = P(\mathcal{R}_I^l = \emptyset) \cdot H_m + (1 - P(\mathcal{R}_I^l = \emptyset)) \cdot H'(T|o_l) \tag{9}$$

H_m denotes the entropy when all linkable queries are dummies. In this case, the linkable queries cannot provide any information about T. However, the adversary can observe all (concurrent) malicious target nodes, and T has chance f to be one of them. Therefore, we have:

$$H_m = (1 - f) \cdot \log_2((1 - f) \cdot N) + f \cdot \log_2(\#mal\_targets) \tag{10}$$

When $\mathcal{R}_I^l$ is non-empty, a range estimation attack based on $\mathcal{R}_I^l$ can produce a minimum range of T. Whereas, an estimation range calculated using any dummy query will be incorrect. Due to use of anonymous paths, an individual dummy query is indistinguishable from any non-dummy query. Nevertheless, the adversary can base on timing and location relationships between queries to filter out some subsets of $\mathcal{Q}_I^l$ that contain dummy queries. In particular, any subset of queries that violates the following rules must contain at least one dummy query:

- if E_i is queried before E_j, then E_i must precede E_j
- if E_i and E_j are the first and last queried nodes in the subset, then any other query must be on the path of the *virtual lookup* from E_i to E_j

Note that the above approach cannot remove all subsets that contain dummy queries. Let $\mathcal{S}_I$ denote all subsets of $\mathcal{Q}_I^l$ that pass the above filtering test, $\mathcal{S}_I \subseteq 2^{\mathcal{Q}_I^l}$. Since all queries in $\mathcal{R}_I^l$ are non-dummy, $\mathcal{R}_I^l$ will pass the filtering test, i.e., $\mathcal{R}_I^l \in \mathcal{S}_I$. From the adversary's prospective, each element of $\mathcal{S}_I$ is possible to be $\mathcal{R}_I^l$. The best strategy for her is to assign different probability to each element in $\mathcal{S}_I$ according to the pre-calculated probability distribution of $\mathcal{R}_I^l$. We use two variables to characterize $\mathcal{R}_I^l$: the number of queries in $\mathcal{R}_I^l$, and the largest hop in the virtual lookup from the first query in $\mathcal{R}_I^l$ to the last query.[6]

Let X denote an arbitrary node that is contained in any estimation range. Then:

$$H'(T|o_l) = -\sum_X P(X = T|o_l) \cdot \log_2 P(X = T|o_l) \tag{11}$$

Let $G(s)$ denote the estimation range computed based on s, $s \in \mathcal{S}_I$. Let $loc(G(s), X)$ denote the location of X in the estimation range $G(s)$, and $|\cdot|$ denote the number of elements of a set. Then, we have:

$$P(X = T|o_l) = \sum_{s \in \mathcal{S}_I} P(s = \mathcal{R}_I^l|o_l) \cdot \gamma(loc(G(s), X), |G(s)|) \tag{12}$$

Let $V(s)$ denote the largest hop in the virtual lookup based on s, and $\chi(x, y)$ denote the probability that a set of x queries with the largest hop in the virtual lookup being y is $\mathcal{R}_I^l$. Then, we have:

$$P(s = \mathcal{R}_I^l|o_l) \approx \frac{\chi(|s|, V(s))}{\sum_{s' \in \mathcal{S}_I} \chi(|s'|, V(s'))} \tag{13}$$

$\chi(x, y)$ is obtained by pre-simulations of the lookup.

Calculation of $H(T|o_d)$. There are three possible cases when there is no linkable query:

- $case_1$: there is no query observed by the adversary
- $case_2$: there is at least one (observed) query that is linkable to B

[6]The largest hop means the largest ID difference between two consecutively queried nodes. The largest hop also implies the number of hops in the lookup. These two characteristics are a close approximation of the adversary's observation on $\mathcal{R}_I^l$.

• $case_3$: there is no query linkable to B but at least one query is observed by the adversary

Let H_1, H_2, and H_3 denote the entropy of T in the three cases, respectively. Then, we have:

$$H(T|o_d) = P(case_1) \cdot H_1 + P(case_2) \cdot H_2 + P(case_3) \cdot H_3 \tag{14}$$

In the first case, since no information is learnt from queries, H_1 is calculated as Equation (10).

For the second case, although I is disassociated with any observed queries, the adversary can group queries belonging to the same lookup based on whether they are linkable to a common relay B. Furthermore, she can calculate estimation ranges for each concurrent lookup that contains queries linkable to B, and consider all estimation ranges as possible candidates for the true estimation range of T.

In particular, we let $\mathcal{R}_I^B$ denote the set of non-dummy queries linkable to B in the lookup performed by I, and H_2 can be calculated as follows.

$$H_2 = P(\mathcal{R}_I^B = \emptyset) \cdot H_m + (1 - P(\mathcal{R}_I^B = \emptyset)) \cdot H_2' \tag{15}$$

where H_2' denotes the entropy when there is at least one non-dummy query linkable to B. If T is malicious, the adversary can reduce the candidates of T down to the set of observed malicious targets; otherwise, she needs to rely on the queries linkable to B to infer T. Therefore, we have:

$$H_2' = f \cdot \log_2(\#mal\_targets) - (1 - f) \cdot \sum_X P(X = T|o_d) \cdot \log_2 P(X = T|o_d) \tag{16}$$

Let Ψ^B denote the set of concurrent lookups that have at least one query linkable to B, and ψ_I denote the lookup performed by I. Let $\mathcal{R}_\psi^B$ denote the set of non-dummy queries linkable to B in ψ, $\psi \in \Psi^B$, and let $\mathcal{S}_\psi$ denote all subsets of queries (linked to B) in ψ that pass the filtering test. Then, we have:

$$P(X = T|o_d) = \sum_{\psi \in \Psi^B} P(\psi = \psi_I|o_d) \cdot \sum_{s \in \mathcal{S}_\psi} P(s = \mathcal{R}_\psi^B|o_d) \cdot \gamma(loc(G(s), X), |G(s)|)$$

Since I is unlinkable to any observed queries, each concurrent lookup is equally likely to be ψ_I. We have:

$$P(\psi = \psi_I|o_d) = \frac{1}{|\Psi^B|}, \quad \forall \psi \in \Psi^B \tag{17}$$

$P(s = \mathcal{R}_\psi^l|o_d)$ is calculated the same as Equation (13).

In the last case, since all observed queries are disassociated with each other, the range estimation attack cannot be applied. Let $\mathcal{R}_I^o$ denote the set of observed non-dummy queries in the lookup performed by I. Similar to the above cases, we have:

$$H_3 = P(\mathcal{R}_I^o = \emptyset) \cdot H_m + (1 - P(\mathcal{R}_I^o = \emptyset)) \cdot H_3' \tag{18}$$

$$H_3' = f \cdot \log_2(\#mal\_targets) - (1 - f) \cdot \sum_X P(X = T|o_d) \cdot \log_2 P(X = T|o_d) \tag{19}$$

Let E_I denote the query closest to T in $\mathcal{R}_I^o$. Then, the adversary can use E_I's successor as the lower bound of the estimation range of T and E_I's predecessor as the upper bound, and assign probabilities to the nodes in the range according to a pre-calculated probability distribution of T. Let $\mathcal{Q}^o$ denote the set of observed queries of all concurrent lookups, and $G(E)$ denote the estimation range based on E. Then, we have:

$$P(X = T|o_d) = \sum_{E \in \mathcal{Q}^o} P(E = E_I|o_d) \cdot \gamma(loc(G(E), X), N - 1) \tag{20}$$

Based on the observations, the adversary is unable to tell which query is more likely to be E_I. Therefore, we have:

$$P(E = E_I|o_d) = \frac{1}{|\mathcal{Q}^o|}, \quad \forall E \in \mathcal{Q}^o \tag{21}$$